Engineering
Software Analytics Studies

Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
http://www.spinellis.gr/
@CoolSWEng
curiosity
serendipity
Outline

• Data gathering
• Processing
  – Methods
  – Advice
• Performance

Data gathering
Alliances

Defining Mobile App Quality
Use BigData analysis to support Android, iOS, HTML5, WP and Windows 8 apps
Organizational adoption of open source software

Dionisios Spinellis a, *, Vaggelis Gannitras b

aDepartment of Management Science and Technology, Athens University of Economics and Business, Patission 76, GR-106 36 Athens, Greece
bInstitute for Manufacturing, University of Cambridge, 17 Charles Babbage Road, Cambridge CB3 0FS, United Kingdom

ARTICLE INFO
Article history:
Received 9 February 2011
Revised 9 March 2011
Accepted 15 September 2011
Available online 21 September 2011
Keywords:
Open source software
Technology adoption
Industrial practice

ABSTRACT
Organizations and individuals can use open source software (OSS) for free, they can study its internal workings, and they can even fix it or modify it to make it suit their particular needs. These attributes make OSS an enticing technological choice for a company. Unfortunately, because most enterprises view technology as a proprietary differentiating element of their operations, little is known about the extent of OSS adoption in industry and the key drivers behind adoption decisions. In this article we examine factors and behaviors associated with the adoption of OSS and provide empirical findings through data gathered from the US Fortune 1000 companies. The data come from each company’s web browsing and browsing activities gathered by sifting through more than 478 million web server logs records and analyzing the results of thousands of network probes. We show that the adoption of OSS is large US companies is significant and is increasing over time through a long-churn transition, advancing from applications to platform. Its adoption is a pragmatic decision influenced by network effects. It isobar in larger organizations and those with many productive employees, and is associated with IT and knowledge-intensive work and operating efficiencies.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

allow the creation of derived works provided they respect the re-use of the derived works. You may not use the derived works for any purpose other than private study.

9
Instrumentation
_MCOUNT_DECL(frompc, selfpc) /* _mcounf; may be static, inline, etc */

u_long frompc, selfpc;
{
    struct gmonparam *p;
    void *stack = &frompc;

    p = &gmonparam;
    /* check that we are profiling
    * and that we aren't recursively invoked.
    */
    if (p->state != GMON_PROF_ON)
        return;
    p->state = GMON_PROF_BUSY;
    /* check that frompcindex is a reasonable pc value.
    * For example: signal catchers get called from the stack,
    * not from text space.  Too bad.
    */
    frompc = p->lowpc;
    if (frompc <= p->textsize)
        write(fd, &stack, sizeof(stack));
    p->state = GMON_PROF_ON;
    return;
}

_MCOUNF
4266 utimensat(AT_FDCWD, 
"boost_1_44_0/libs/bind/test/bind_stdcall_mf_test.cpp", {{1336157319, 
67677176}, {1090801932, 0}}, 0) = 0
4266 chown("boost_1_44_0/libs/bind/test/bind_stdcall_mf_test.cpp", 0, 0) = 0
4266 chmod("boost_1_44_0/libs/bind/test/bind_stdcall_mf_test.cpp", 0644) = 0
4266 open("boost_1_44_0/libs/bind/test/bind_stdcall_test.cpp", 
O_WRONLY|O_CREAT|O_EXCL, 0600) = 4
4266 write(4, "...", 2789) = 2789
4266 close(4)
4266 utimensat(AT_FDCWD, 
"boost_1_44_0/libs/bind/test/bind_stdcall_mf_test.cpp", {{1336157319, 
67677176}, {1090801932, 0}}, 0) = 0
4266 chown("boost_1_44_0/libs/bind/test/bind_stdcall_mf_test.cpp", 0, 0) = 0
4266 chmod("boost_1_44_0/libs/bind/test/bind_stdcall_test.cpp", 0644) = 0
4266 open("boost_1_44_0/libs/bind/test/bind_test.cpp", 
O_WRONLY|O_CREAT|O_EXCL, 0600) = 4
4266 read(3, <unfinished ...>
Recursion
GHTorrent: Github’s Data From a Firehose

Georgios Gousios and Dionisios Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
Athens, Greece
{gousiosg,ds}@aueb.gr

Abstract—A common requirement of many empirical software engineering studies is the acquisition and curation of data from software repositories. During the last few years, GitHub has emerged as a popular project hosting, mirroring, and collaboration platform. GitHub provides an extensive REST API, which enables researchers to retrieve both the commits to the projects’ repositories and events generated through user actions on project resources. GHTorrent aims to create a scalable offline mirror of GitHub’s event streams and persistent data, and offer it to the research community as a service. In this paper, we present the project’s design and initial implementation and demonstrate how the provided datasets can be queried and processed.

Keywords—dataset; repository; GitHub; commits; events

1. INTRODUCTION AND MOTIVATION

When conducting empirical research with data from software repositories, a typical step involves the downloading of the project data to be analysed. Obtaining data from open source software (OSS) project repositories is a tedious exercise lacking scientific value, while the obtained datasets are often non-homogeneous which makes further analysis also facilitates project contributions from non-team member through features such as forking, which creates a private copy of a git repository, and pull requests, which organize a series of commits to a forked repository into a mergeable patch for the source repository. Moreover, GitHub offers the usual gamut of software forge facilities, such as bug trackers, a wiki, and developer communication tools. What makes GitHub particularly attractive for the repository mining community, is that it provides access to its internal data stores through an extensive REST API, which researchers can use to access a rich collection of metrics and versioned process and product data.

Although GitHub’s API is well designed and documented, researchers wishing to use it in order to analyze its data will face a number of challenges:

• GitHub’s data is huge (of the terabyte scale according to our estimation) and an API rate limit of five thousand requests per hour makes the complete download of the data impossible.
• The overall schema of GitHub’s data is not documented, and there are a variety of reasons to believe it would be...
About CHABADA

How do we know a program does what it claims to do? After clustering Android apps by their description topics, we identify outliers in each cluster with respect to their API usage. A "weather" app that sends messages that becomes an anomaly; likewise, a "messaging" app would typically not be expected to access the current location. Applied on a set of 2,500+ Android applications, our CHABADA prototype identified several anomalies, additionally it flagged 56% of novel malwares such, without requiring any known malware patterns.

Videos

- Watch the video of the CHABADA talk presented at Microsoft Research.
- CHABADA on ARD: Watch the video here (in german)
- CHABADA on ZDF: Watch the video here (in german)

Downloads

- Download the preprint of the CHABADA paper here. The final revised version of the paper will appear at the International Conference of Software Engineering (ICSE) 2014.
- Download the whole dataset of the CHABADA experiments here.

@seppgather/CISBenach/CHABADA_ICSE_2014
author = Alessandra Gori and Berke Teweschi and Florian Groes and Andreas Zeller,
title = (Checking App Behavior Against App Descriptions),
booktitle = ICSE '14: Proceedings of the 36th International Conference on Software Engineering,
location = Hyderabad (India), 31 May - 7 June,
year = 2014.

[Image of CHABADA website]

[Image of app store with similar apps and developer profile]
git fast-import
$ git log --reverse --date-order
commit 94e213242a65085bb258eee32a4a15f8b351f73f9d7a1
Author: Ken Thompson and Dennis Ritchie <research!ken,dom>
Date: Tue Jun 20 08:00:00 1972 -0800
Research V1 development
Work on file u5.s
commit 7f62ab480a397066463a8fd975a8707923d0349
[-]
Author: Ken Thompson <research!ken>
Date: Tue Nov 26 18:13:21 1974 -0500
Research V5 development
Work on file sys/kern/alp.c
commit 820967416a7f4577b4101802bb8e732956d249
[-]
Author: Dennis Ritchie <research!dr
Date: Mon Dec 2 18:18:02 1974 -0500
Research V5 development
Work on file sys/dar/kb.c
commit 71393393f8e4a02969ce260bdecac433a8504f9
Author: Brian W. Kernighan <research!bk>
Date: Tue May 11 10:43:47 1975 -0500
Research V5 development
Work on file sys/dar/kb.c
commit acb97b04634a46d87005f10247970112838360
Author: S. R. Bourne <research!sb
Date: Fri Jan 12 02:17:45 1979 -0500
Research V7 development
Work on file src/cmd/sh/blok.c
commit eeb0b0d0b936e65b64f38702d5a7a68d7
Author: Eric Schmidt <schmidt@ucbvax.Berkeley.EDU>
Date: Sat Jan 5 22:49:18 1980 -0800
BSD 3 development
Work on file src/cmd/nt/sub.c

$ git blame -C -C src/sys/pipe.c
1c1008e src/sys/kern/pipe.c (Ken Thompson 1974-11-26 18:13:23 -0800 53) rf->f_flag = FREAD|FPIPE;
1c1008e src/sys/kern/pipe.c (Ken Thompson 1974-11-26 18:13:23 -0800 54) rf->f_inode = ip;
1c1008e src/sys/kern/pipe.c (Ken Thompson 1974-11-26 18:13:23 -0800 55) ip->s_count = 2;
[...]
1f8302e src/sys/pipe.c (Ken Thompson 1979-05-18 15:19:33 -0800 125) register struct inode *ip;
1f8302e src/sys/pipe.c (Ken Thompson 1979-05-18 15:19:33 -0800 126) ip = ip->i_inode;
1f8302e src/sys/pipe.c (Ken Thompson 1979-05-18 15:19:33 -0800 127) c = u.u_count;
1f8302e src/sys/pipe.c (Ken Thompson 1979-05-18 15:19:33 -0800 128) lo:";
1f8302e src/sys/pipe.c (Ken Thompson 1979-05-18 15:19:33 -0800 129) /*
9a9f62e src/sys/pipe.c (Bill Joy 1980-01-05 05:11:16 -0800 130) * If error or all done, return.
9a9f62e src/sys/pipe.c (Bill Joy 1980-01-05 05:11:16 -0800 131) */
9a9f62e src/sys/pipe.c (Bill Joy 1980-01-05 05:11:16 -0800 132) if (u.u_error)
9a9f62e src/sys/pipe.c (Bill Joy 1980-01-05 05:11:16 -0800 133) return;
6d62e8e src/sys/pipe.c (Ken Thompson 1975-07-17 18:10:37 -0800 135) plste(p); if(x = W) {
6d62e8e src/sys/pipe.c (Ken Thompson 1975-07-17 18:10:37 -0800 136) prele(p);
6d62e8e src/sys/pipe.c (Ken Thompson 1975-07-17 18:10:37 -0800 137) return;
6d62e8e src/sys/pipe.c (Ken Thompson 1975-07-17 18:10:37 -0800 138) u.i_count = 0;
6d62e8e src/sys/pipe.c (Ken Thompson 1975-07-17 18:10:37 -0800 139) return;
[...]
Probing

nmap
Processing methods
Count word occurrences

• 20GB in 36,260 files
• Hadoop: 12:25:20
Count word occurrences

- 20GB in 36,260 files
- Hadoop: 12:25:20
- sgsh: 00:34:38
find Gutenberg -type f | xargs cat | tr -s ' \t\n\r\f' ' \n | sort -S 6G | uniq -c

Count word occurrences

• 20GB in 36,260 files
• Hadoop: 12:25:20
• sgsh: 00:34:38
• Unix pipeline: 00:21:37
Processing pipelines

Get
Select

Process
curl
wget
git log
git blame
$ # Print author names
git log --format='%an' |

# Order them by author
sort |

# Count number of commits for each author
uniq --count |

# Order them by number of commits
sort --numeric-sort --reverse |

# Print top ten
head -10

20131 Linus Torvalds
8445 David S. Miller
7692 Andrew Morton
5156 Greg Kroah-Hartman
5116 Mark Brown
4723 Russell King
4584 Takashi Iwai
4385 Al Viro
4220 Ingo Molnar
3276 Tejun Heo
Division of Effort, Productivity, Quality, and Relationships in FLOSS Virtual Teams: Evidence from the FreeBSD Project

George M. Giaglis
(Department of Management Science and Technology, Athens University of Economics and Business, Greece
giaglis@aeub.gr)

Dionisis Spinellis
(Department of Management Science and Technology, Athens University of Economics and Business, Greece
dds@aeub.gr)

Abstract: Research on virtual teams and distributed work argues that the lack of co-location places an overhead on the performance potential of large, globally distributed teams. In this paper, we revisit this tenet through a case study of FreeBSD Open Source Software (FLOSS) development to demonstrate how globally dispersed FLOSS communities manage to overcome the problem of geographic separation of their members. Our results show that successful FLOSS teams demonstrate a truly global distribution of members, who perform different types of work so as to achieve consistent round-the-clock development, without any apparent ill effects on team productivity and the quality of the resulting outcomes. Cooperation between team members is abundant, especially at more complex work items, and does not seem to be affected by distance. Only monitoring relationships appear in some cases to be easier to cultivate between individuals living closer together. These findings challenge the conventional wisdom of research in distributed work, in cases where virtual teams consist of highly skilled and motivated individuals who leverage the power of communication technologies to overcome problems associated with physical distance.

Keywords: FLOSS, open source development, virtual teams, distributed work, FreeBSD
Categories: D.2.9, K.4.3, K.6.4, K.6.5
grep
cut
awk
sed

Process
sort

-\( n \)
-\( r \)
+\( 2nr \)
-\( u \)
-\( t: \)
diff

$ diff file1 file2 | grep '^[<>]' | wc -l
<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch</td>
<td>bash</td>
<td>cat</td>
<td>chflags</td>
</tr>
<tr>
<td>chgrp</td>
<td>chio</td>
<td>chmod</td>
<td>cp</td>
</tr>
<tr>
<td>chown</td>
<td>cpio</td>
<td>csh</td>
<td>date</td>
</tr>
<tr>
<td>dd</td>
<td>df</td>
<td>echo</td>
<td>ed</td>
</tr>
<tr>
<td>dir</td>
<td>dmesg</td>
<td>domainname</td>
<td>echo</td>
</tr>
<tr>
<td>dnsdomainname</td>
<td></td>
<td></td>
<td>ed</td>
</tr>
<tr>
<td>egrep</td>
<td>expr</td>
<td>getfacl</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$ comm Linux FreeBSD
join

gvpr
digraph D {
    "archivers/arj" -> "devel/gmake"
    "archivers/arj" -> "devel/autoconf259"
    "archivers/dact" -> "devel/gmake"
    "archivers/dact" -> "security/libmcrypt"
    "archivers/dact" -> "archivers/lzo"
    "archivers/deepforest" -> "x11-toolkits/tkstep80"
    "archivers/deepforest" -> "graphics/libimg"
    "archivers/deepforest" -> "x11/xorg-libraries"
    "archivers/fastjar" -> "devel/gmake"
    "archivers/fastjar" -> "lang/perl5.8"
    "archivers/gtar" -> "converters/libiconv"
    "archivers/gtar" -> "devel/gettext"
}

BEG_G {
    $tvtype = TV_fwd;
    $tvroot = node($, ARGV[0]);
}
N [$tvroot = NULL; 1]
END_G {
    induce($T);
    write($T);
    exit(0);
}
Open Source Licensing across Package Dependencies

Maria Kechagia, Dionysios Spinellis and Stephanois Andreasellis-Theotokis
Department of Management Science and Technology
Athens University of Economics and Business
Athens, Greece GR-104 34
Email: mkelagia@dsnt.aueb.gr, {ddks, stheotok}@aueb.gr

Abstract—Licensing dependencies among open source software (OSS) packages reveal a lot about software compatibility relationships and the practicabilities of OSS licensing. There is, however, limited information on these in the literature. In this paper, we discuss various aspects of OSS licensing, and present an empirical study on FreeBSD ports collections concerning their licensing dependencies, in an attempt to identify specific patterns. Our results highlight different types of dependencies, that could be used to explain, or even guide the license selection process of OSS projects.

Keywords—OSS, licenses; dependencies; FreeBSD ports

I. INTRODUCTION

Open source software (OSS), can be freely used, modified, and distributed, provided certain restrictions are observed regarding its copyright and the protection of its status as OSS. These rights and restrictions are expressed through the software’s license, i.e. a contract between the software owners (the licensor) and its prospective users (the licensees). OSS

1) Are some license types particularly conducive to reuse?
2) Do the licenses across dependencies follow an order associated with the permissiveness of each license?

Results could explain or even guide the selection of a particular OSS license.

II. CONCEPTS AND DEFINITIONS

Before discussing the different types of OSS licenses, we briefly introduce the background concepts delineating the degrees of freedom available while distributing the product of any intellectual activity, including software.

A. Intellectual property, copyrights and patents

The term intellectual property is used to encompass a wide range of areas of law, including copyrights, patents and even trademarks [2]. These are all means used to encourage private investment in research, technology and innovation, by ensuring that innovators will be able to get private returns for their work.
Summarize
wc
uniq
head
tail
fmt

awk
awk '{print $2}' licenses | sort | uniq -c | sort -rn
sgsh — scatter-gather shell

The scatter-gather shell, sgsh, provides an expressive way to construct sophisticated and efficient big data sets and stream processing pipelines using existing Unix tools as well as custom-built components. It is a Unix-style shell allowing the specification of pipelines with non-linear scatter-gather operations. These form a directed acyclic process graph, which is typically executed by multiple processor cores, thus increasing the operation's processing throughput.

If you want to get a feeling on how sgsh works in practice, skip right down to the examples section.

Inter-process communication

Sgsh provides three new ways for expressing inter-process communication.

**Scatter blocks**

Send the output of one pipeline ending with `[]` into multiple pipelines beginning with `[]`. The scatter block is terminated by `[]`. Scatter interconnections are internally implemented through automatically-named pipes, and a helper program, `sgsh-scatter()`, that distributes the data to multiple processes. The scatter behavior can be modified by adding `sgsh-scatter()` flags after the `[]` symbol.

**Stores**

Are named as `store_name`. These allow the storage of a data stream's last second (or of a specified window of records) into a named buffer. This record can be later retrieved asynchronously by one or more readers. Data can be pushed into a store or out of a store, or it can be read using the shell's command output substitution syntax. Stores are implemented internally through Unix-domain sockets, a writer program, `sgsh-writer()`, and a reader program, `sgsh-reader()`. The behavior of store I/O can be modified by adding `sgsh-writer()` and `sgsh-reader()` flags after the store's name. In particular, flags can be easily used to operate on windows of stream data, rather than a single value written to a store.
Scripting languages

github.com/dspinellis/unix-history-repo

- Research Edition Unix: V1, V3–V7
- Unix 32V
- BSD 1, 2, 3, 4, 4.1, 4.2, 4.3 *, 4.4 *
- 386BSD 0.0, 0.1
- FreeBSD 1.0–10.0

- Tags
- Contributors
- Branches and merges
Tool front ends

find /usr/ports/ -name Makefile
-maxdepth 3 |
sed 's,/Makefile,,' |
while read dir
do
    cd $dir
    make -V PORTNAME\
    -V RUN_DEPENDS -V BUILD_DEPENDS\
    -V LIB_DEPENDS -V FETCH_DEPENDS\
    -V DEPENDS
done
Analyze Binaries

nm, readelf, ar, ldd, dumpbin, javap
$ cd /usr/bin
$ ldd * 2>/dev/null |
awk '/=>/{print $3}' | sort | uniq -c |
sort -rn | head

541 /lib/libc.so.6
541 /lib/ld-linux.so.2
104 /lib/libdl.so.2
  92 /lib/libm.so.6
  62 /lib/libncurses.so.5
  59 /lib/libnsl.so.1
  50 /lib/libcrypt.so.1
  34 /usr/lib/libstdc++-libc6.2-2.so.3
  34 /lib/libpam.so.0
  28 /usr/lib/libz.so.1
The Bug Catalog of the Maven Ecosystem

Dimitris Mitropoulos* Vassilios Karakoidas* Diomidis Spinellis*
*Department of Management Science and Technology
Athena University of Economics and Business
Athena, Greece
dimitro, bkarak, doidis@ualu.edu

Panos Louridas† Georgios Gousios†
†Software Engineering Research Group
Delft University of Technology
Delft, the Netherlands
gousios@tudelft.nl

ABSTRACT
Examining software ecosystems can provide the research community with data regarding artifacts, processes, and communities. We present a dataset obtained from the Maven central repository ecosystem (approximately 265G of data) by statistically analyzing the repository to detect potential software bugs. For our analysis we used FindBugs, a tool that examines Java bytecode to detect numerous types of bugs. The dataset contains the metric results that FindBugs reports for every project version (a JAR) included in the ecosystem. For every version we also store specific metadata such as the JAR’s size, its dependencies and others. Our dataset can be used to produce interesting research results, as we show in specific examples.

Categories and Subject Descriptors
1.2.4 [Software Engineering]: Software/Program Veri-

Table 1: Descriptive statistic measurements for the Maven repository.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>17505</td>
</tr>
<tr>
<td>Versions (total)</td>
<td>115214</td>
</tr>
<tr>
<td>Min (versions per project)</td>
<td>1</td>
</tr>
<tr>
<td>Max (versions per project)</td>
<td>338</td>
</tr>
<tr>
<td>Mean (versions per project)</td>
<td>6.58</td>
</tr>
<tr>
<td>Median (versions per project)</td>
<td>3</td>
</tr>
<tr>
<td>Range (over versions)</td>
<td>337</td>
</tr>
<tr>
<td>1st Quartile (over versions)</td>
<td>1</td>
</tr>
<tr>
<td>3rd Quartile (over versions)</td>
<td>8</td>
</tr>
</tbody>
</table>
Tool Interoperability
ckjm — Chidamber and Kemerer Java Metrics

The program ckjm calculates Chidamber and Kemerer object-oriented metrics by processing the bytecode of compiled Java files. The program calculates for each class the following six metrics proposed by Chidamber and Kemerer:

- VMC: Weighted methods per class
- DIT: Depth of Inheritance Tree
- NCC: Number of Children
- DCO: Coupling between object classes
- RFC: Response for a Class
- CCOC: Lack of cohesion in methods

In addition, it also calculates for each class:

- Cc. Average couplings
- NPM: Number of public methods

Citation and Background

If you use this tool in your research, please cite it as follows:

Kleisli, R., Spadini, L. (2005). JUnit: The Open Source Perspective. The programs I found on the web were either incomplete (they calculated only some of the metrics) or unreliable (their calculated results that were obviously wrong, or extremely unlikely they required less than 1.8MB and block of processes in Eclipse) because of the size. However, after trying and testing a 1.4GHz Pentium-M machine it still processes the 1.8MB of the Eclipse 3.0 jar files (1717 classes) in 9 seconds.
http://geonames.usgs.gov/
http://earth-info.nga.mil/
The Carbon Footprint of Conference Papers

Dimitris Spinellis 1, Panos Louillard 1,2,3
1 Department of Management Science and Technology, Athens University of Economics and Business, Athens, Greece, 2 Research and Technology Network, Athens, Greece

Abstract

The action required to stem the environmental and social implications of climate change depends crucially on how human activities shape technology, economy, lifestyle and policy. With transport CO2 emissions accounting for about a quarter of the total, we examine the contribution of CO2 output by scientific travel. Thankfully for the reputation of the scientific community, CO2 emissions associated with the trip required for the paper are at scientific conference account for just 0.003% of the yearly total. However, with CO2 emissions for a single conference trip amounting to 7% of an average individual’s total CO2 emissions, scientists should lead by example by demonstrating leadership in addressing this issue.

Introduction

The environmental and social implications of climate change depend not only on Earth’s economic responses but also on how human activities shape technology, economy, lifestyle and policy.1 Action should not be postponed, as it is argued that we have already surpassed a safe threshold in atmospheric carbon dioxide concentration (from a 288 ppm pre-industrial value to 397 ppm today, with a projected boundary threshold of 550 ppm).2 Changes in economy, lifestyle, and policy, must change in human behaviour, which will ultimately require decisions involving moral judgements. Decisions should not be put off, considering that other considerations may be from the same order of magnitude as the one above.

1 E-mail: louillard@uoa.gr

Compliance with ethical standards:
No institutional or national research ethics committee approval was required for this manuscript.

Financial support: The authors have declared that no competing interests exist.

1 54
Processing advice
AITMOAF

Automate
make
set -e
a && b && c
top/htop
logger
pv
monitor

(www.spinellis.gr/blog/20081027/)

Take notes
Checkpoint

Handle broken connections
nohup
mosh

Performance
Optimize

Parallelize
parallel
find . -name ".c" -type f -print0 | xargs -0 -n 1 \
   analyze-file.sh

find . -name "*.c" -type f -print0 | parallel --keep-order -d '\0' -n 1 -gnu \
   analyze-file.sh
Low-level processing

The Collaborative Organization of Knowledge

Why Wikipedia’s remarkable growth is sustainable.

BY DIONIDIS SPINELLIS AND PANAGIOTIS LOURIDAS

snapshot of all recorded changes and examined how entries are created and linked. Inspecting the timestamps on individual entry definitions and references, we found that links to nonexistent articles often precede creation of new articles. Also, tracking the evolution of article links allowed us to empirically validate Barabasi’s hypothesis on the formation of scale-free graphs through incremental growth and preferential attachment. Our findings paint a picture of sustainable growth, suggesting that Wikipedia’s development process delivers coverage of more and more subjects.

The phenomenal growth of Wikipedia is attributable to a mixture of technologies and a process of open participation. The key technology behind Wikipedia is that of a Wiki—online lightweight Web-based collaborations. Wikipedia content appears online as static HTML pages, though each such page includes an edit button anyone can use to modify its content; editing most articles requires no prior author-

contributed articles
class RangeMap {
private:
    vector<bool> active;
public:
    static const int NMONTH = (2009 - 2001) * 12;
    RangeMap() : active(NMONTH, false) {}
};

class EntryDetails {
private:
    RangeMap defined, stub;
    vector<int> nrefs;
    string definer, referrer, firstRef;
    time_t firstDef;
    int numReferences, numContributors;
    int numRevisions, numReverts;
public:
    EntryDetails() : defined(), stub(),
    nrefs(RangeMap::NMONTH, 0), firstDef(-1),
    numReferences(0), numContributors(0), numRevisions(0),
    numReverts(0) {}
};
class RangeMap {
    private BitSet active;
    public static final int NMONTH = (2009 - 2001) * 12;
    public RangeMap() {
        active = new BitSet(NMONTH);
    }
}

class EntryDetails {
    RangeMap defined, stub;
    ArrayList<Integer> nrefs;
    String definer, referrer, firstRef;
    Date firstDef;
    int numReferences, numContributors;
    int numRevisions, numReverts;
    public EntryDetails() {
        defined = new RangeMap();
        stub = new RangeMap();
        nrefs = new ArrayList<Integer>(RangeMap.NMONTH);
    }
}

12,287
7,212
C++
Java
Experiment on a subset

Sample
Thank you!

http://www.spinellis.gr
@CoolSWEng
dds@aueb.gr

Image Credits

- Aero Icarus
- Chrysler Group
- Jose Maria Minarro Vivancos
- Kirti Poddar
- Lab Science Career
- Lachlan Donald
- Luigi Chiesa
- Marco40134
- Mark Sze
- Meats Meyer
- NASA
- NTSB
- PSParrot
- Q-Wing
- Rachel Kramer
- Sam Rayner
- Tim Bouwer
- Tudor Barker
- Wikimedia Commons
- William Avery, W. Heath Robinson
- kaktuslampan
- kenmainr
- max guitare
- patrasTimes.gr
- vlc team, ubuntu, Hidro
- zzpza